The Theory of WSTS: The Case of Complete WSTS*

Alain Finkel and Jean Goubault-Larrecq

ENS Cachan {finkel,goubault}@lsv.ens-cachan.fr

Abstract. We describe a simple, conceptual forward analysis procedure for ∞-complete WSTS $\mathfrak{S}$. This computes the so-called *clover* of a state. When $\mathfrak{S}$ is the completion of a WSTS $\mathfrak{X}$, the clover in $\mathfrak{S}$ is a finite description of the downward closure of the reachability set. We show that such completions are ∞-complete exactly when $\mathfrak{X}$ is an ω^2-*WSTS*, a new robust class of WSTS. We show that our procedure terminates in more cases than the generalized Karp-Miller procedure on extensions of Petri nets. We characterize the WSTS where our procedure terminates as those that are *clover-flattable*. Finally, we apply this to well-structured Presburger counter systems.

1 Introduction

Context. Well-structured transition systems (WSTS) [Fin87,Fin90,FS01,AČJT00] are a general class of infinite-state systems where coverability—given states s, t, decide whether $s \geq s_1 \to^* t_1 \geq t$ for some s_1, t_1—is decidable, using a simple algorithm that works backwards. The starting point of the series of papers entitled *Forward analysis for WSTS, part I: Completions* [FG09a], and *Forward analysis for WSTS, part II: Complete WSTS* [FG09b] Simplis our desire to derive similar algorithms working *forwards*, namely algorithms computing the *cover* $\downarrow Post^*(\downarrow s)$ of s. While the cover allows one to decide coverability as well, by testing whether $t \in \downarrow Post^*(\downarrow s)$, it can also be used to decide the boundedness problem, i.e., to decide whether the reachability set, $Post^*(s)$, is finite. No backward algorithm can decide this. In fact, boundedness is undecidable in general, e.g., on reset Petri nets [DFS98]. So the reader should be warned that computing the cover is not possible for general WSTS. Despite this, the known forward algorithms are felt to be more efficient than backward procedures in general: e.g., for lossy channel systems, although the backward procedure always terminates, only a (necessarily non-terminating) forward procedure is implemented in the TREX tool [ABJ98]. Another argument in favor of forward procedures is the following: for depth-bounded processes, a fragment of the π-calculus, the backward algorithm of [AČJT00] is not applicable when the maximal depth of configurations is not known in advance because, in this case, the predecessor configurations are not effectively computable [WZH10]. But the *forward* Expand,

* This paper is an extended abstract of a complete paper that will appear in the journal LMCS [FG12b]. A short version has already appeared in [FG09b] at ICALP'09.

Enlarge and Check algorithm of [GRvB07], which operates on complete WSTS, solves coverability even though the depth of the process is not known a priori [WZH10].

State of the Art. Karp and Miller [KM69] proposed an algorithm, for Petri nets, which computes a finite representation of the *cover*, i.e., of the downward closure of the reachability set of a Petri net. Finkel [Fin87,Fin90] introduced the framework of WSTS and generalized the Karp-Miller procedure to a class of WSTS. This was achieved by building a non-effective completion of the set of states, and replacing ω-accelerations of increasing sequences of states (in Petri nets) by least upper bounds. In [EN98,Fin90] a variant of this generalization of the Karp-Miller procedure was studied; but no guarantee was given that the cover could be represented finitely. In fact, no effective finite representations of downward-closed sets were given in [Fin90]. Finkel [Fin93] modified the Karp-Miller algorithm to reduce the size of the intermediate computed trees. Geeraerts *et al.* [GRvB07] recently proposed a weaker acceleration, which avoids some possible underapproximations in [Fin93]. Emerson and Namjoshi [EN98] take into account the labeling of WSTS and consequently adapt the generalized Karp-Miller algorithm to model-checking. They assume the existence of a compatible dcpo, and generalize the Karp-Miller procedure to the case of broadcast protocols (which are equivalent to transfer Petri nets). However, termination is then not guaranteed [EFM99], and in fact neither is the existence of a finite representation of the cover. We solved the latter problem in [FG09a].

Abdulla, Collomb-Annichini, Bouajjani and Jonsson proposed a forward procedure for lossy channel systems [ACABJ04] using downward-closed regular languages as symbolic representations. Ganty, Geeraerts, Raskin and Van Begin [GRvB06b,GRvB06a] proposed a forward procedure for solving the coverability problem for WSTS equipped with an effective adequate domain of limits, or equipped with a finite set D used as a parameter to tune the precision of an abstract domain. Both solutions ensure that every downward-closed set has a finite representation. Abdulla *et al.* [ACABJ04] applied this framework to Petri nets and lossy channel systems. Abdulla, Deneux, Mahata and Nylén proposed a symbolic framework for dealing with downward-closed sets for Timed Petri nets [ADMN04].

Our Contribution. First, we define a *complete WSTS* as a WSTS $\mathfrak{S}$ whose well-ordering is also a continuous dcpo (a dcpo is a directed complete partial ordering). This allows us to design a conceptual procedure $\mathbf{Clover}_{\mathfrak{S}}$ that looks for a finite representation of the downward closure of the reachability set, i.e., of the cover [Fin90]. We call such a finite representation a *clover* (for *clo*sure of *cover*). This clearly separates the fundamental ideas from the data structures used in implementing Karp-Miller-like algorithms. Our procedure also terminates in more cases than the well-known (generalized) Karp-Miller procedure [EN98,Fin90]. We establish the main properties of clovers in Section 4 and use them to prove $\mathbf{Clover}_{\mathfrak{S}}$ correct, notably, in Section 6.

Second, we characterize complete WSTS for which $\mathbf{Clover}_{\mathfrak{S}}$ terminates. These are the ones that have a (continuous) flattening with the same clover. This establishes a surprising relationship with the theory of flattening [BFLS05]. The result (Theorem 7), together with its corollary on covers, rather than clovers (Theorem 8), is the main achievement of this paper.

Third, and building on our theory of completions [FG09a], we characterize those WSTS whose completion is a complete WSTS in the sense above. They are exactly the ω^2-*WSTS*, i.e., those whose state space is ω^2-wqo (a wqo is a well quasi-ordering), as we show in Section 5. All naturally occurring WSTS are in fact ω^2-WSTS. We shall also explain why this study is important: despite the fact that $\mathbf{Clover}_{\mathfrak{S}}$ cannot terminate on all inputs, that $\mathfrak{S}$ is an ω^2-WSTS will ensure *progress*, i.e., that every opportunity of accelerating a loop will eventually be taken by $\mathbf{Clover}_{\mathfrak{S}}$.

Finally, we apply our framework of complete WSTS to counter systems in Section 7. We show that affine counter systems may be completed into ∞-complete WSTS iff the domains of the monotonic affine functions are upward-closed.

2 Preliminaries

We borrow from theories of order, as used in model-checking [AČJT00,FS01], and also from domain theory [AJ94,GHK$^+$03]. A *quasi-ordering* $\leq$ is a reflexive and transitive relation on a set X. It is a (partial) *ordering* iff it is antisymmetric.

We write $<$ for the associated strict ordering ($\leq \setminus \geq$), There is also an associated equivalence relation $\equiv$, defined as $\leq \cap \geq$.

A set X with a partial ordering $\leq$ is a *poset* $(X, \leq)$, or just X when $\leq$ is clear. If X is merely quasi-ordered by $\leq$, then the quotient $X/\equiv$ is ordered by the relation induced by $\leq$ on equivalence classes. So there is not much difference in dealing with quasi-orderings or partial orderings, and we shall essentially be concerned with the latter.

The *upward closure* $\uparrow E$ of a set E in X is $\{y \in X \mid \exists x \in E \cdot x \leq y\}$. The *downward closure* $\downarrow E$ is $\{y \in X \mid \exists x \in E \cdot y \leq x\}$. A subset E of X is *upward-closed* if and only if $E = \uparrow E$. *Downward-closed* sets are defined similarly. A *basis* of a downward-closed (resp. upward-closed) set E is a subset A such that $E = \downarrow A$ (resp. $E = \uparrow A$); E has a *finite basis* iff A can be chosen to be finite.

A quasi-ordering is *well* iff from any infinite sequence $x_0, x_1, \ldots, x_i, \ldots$, one can extract an infinite ascending chain $x_{i_0} \leq x_{i_1} \leq \ldots \leq x_{i_k} \leq \ldots$, with $i_0 < i_1 < \ldots < i_k < \ldots$. While *wqo* stands for well-quasi-ordered set, we abbreviate well posets as *wpos*.

An *upper bound* $x \in X$ of $E \subseteq X$ is such that $y \leq x$ for every $y \in E$. The *least upper bound (lub)* of a set E, if it exists, is written $\mathrm{lub}(E)$. An element x of E is *maximal* (resp. minimal) iff $\uparrow x \cap E = \{x\}$ (resp. $\downarrow x \cap E = \{x\}$). Write $\mathrm{Max}\, E$ (resp. $\mathrm{Min}\, E$) for the set of maximal (resp. minimal) elements of E.

A *directed subset* of X is any non-empty subset D such that every pair of elements of D has an upper bound in D. Chains, i.e., totally ordered subsets, and one-element sets are examples of directed subsets. A *dcpo* is a poset in which

every directed subset has a least upper bound. For any subset E of a dcpo X, let $\mathrm{Lub}(E) = \{\mathrm{lub}(D) \mid D \text{ directed subset of } E\}$. Clearly, $E \subseteq \mathrm{Lub}(E)$; $\mathrm{Lub}(E)$ can be thought of E plus all limits from elements of E.

The *way below* relation $\ll$ on a dcpo X is defined by $x \ll y$ iff, for every directed subset D such that $\mathrm{lub}(D) \leq y$, there is a $z \in D$ such that $x \leq z$. Note that $x \ll y$ implies $x \leq y$, and that $x' \leq x \ll y \leq y'$ implies $x' \ll y'$. Write $\mathdownarrow E = \{y \in X \mid \exists x \in E \cdot y \ll x\}$, and $\mathdownarrow x = \mathdownarrow\{x\}$. X is *continuous* iff, for every $x \in X$, $\mathdownarrow x$ is a directed subset, and has x as least upper bound.

When $\leq$ is a well partial ordering that also turns X into a dcpo, we say that X is a *directed complete well order*, or *dcwo*. We shall be particularly interested in continuous dcwos.

A subset U of a dcpo X is (Scott-)*open* iff U is upward-closed, and for any directed subset D of X such that $\mathrm{lub}(D) \in U$, some element of D is already in U. A map $f : X \to X$ is (Scott-)*continuous* iff f is monotonic ($x \leq y$ implies $f(x) \leq f(y)$) and for every directed subset D of X, $\mathrm{lub}(f(D)) = f(\mathrm{lub}(D))$. Equivalently, f is continuous in the topological sense, i.e., $f^{-1}(U)$ is open for every open U.

A weaker requirement is ω-continuity: f is *ω-continuous* iff $\mathrm{lub}\{f(x_n) \mid n \in \mathbb{N}\} = f(\mathrm{lub}\{x_n \mid n \in \mathbb{N}\})$, for every countable chain $(x_n)_{n\in\mathbb{N}}$. This is all we require when we define accelerations, but general continuity is more natural in proofs. We won't discuss this any further: the two notions coincide when X is countable, which will always be the case of the state spaces X we are interested in, where the states should be representable on a Turing machine, hence at most countably many.

The *closed* sets are the complements of open sets. Every closed set is downward-closed. On a dcpo, the closed subsets are the subsets B that are both downward-closed and *inductive*, i.e., such that $\mathrm{Lub}(B) = B$. An inductive subset of X is none other than a sub-dcpo of X.

The *closure* $cl(A)$ of $A \subseteq X$ is the smallest closed set containing A. This should not be confused with the *inductive closure* $\mathrm{Ind}(A)$ of A, which is obtained as the smallest inductive subset B containing A. In general, $\downarrow A \subseteq \mathrm{Lub}(\downarrow A) \subseteq \mathrm{Ind}(\downarrow A) \subseteq cl(A)$, and all inclusions can be strict. All this nitpicking is irrelevant when X is a *continuous* dcpo, and A is downward-closed in X. In this case indeed, $\mathrm{Lub}(A) = \mathrm{Ind}(A) = cl(A)$. This is well-known, see e.g., [FG09a, Proposition 3.5], and will play an important role in our constructions. As a matter in fact, the fact that $\mathrm{Lub}(A) = cl(A)$, in the particular case of continuous dcpos, is required for lub-accelerations to ever reach the closure of the set of states that are reachable in a transition system.

3 A survey on Well-Structured Transition Systems

WSTS were originally thought of as generalizations of Petri nets, in which the set of states (called markings) of a Petri net with n places, $\mathbb{N}^n$, is abstracted into a set X equipped with a wpo $\leq$; the Petri net transitions (which are affine translations from $\mathbb{N}^n$ into $\mathbb{N}^n$) are abstracted to general recursive monotonic

functions from X to X. WSTS were defined and studied in the first author's PhD thesis in 1986, the results were presented at ICALP'87 [Fin87] and published in [Fin90]. The theory of WSTS has now been used for 25 years as a foundation for verification in various models, such as (monotonic extensions of) Petri nets, broadcast protocols, fragments of the pi-calculus fragments, rewriting systems, lossy systems, timed Petri nets, etc.

3.1 Well-Structured Transition Systems: from 1986 to 1996

A *transition system* is a pair $\mathfrak{S} = (S, \rightarrow)$ of a set S, whose elements are called *states*, and a *transition relation* $\rightarrow \subseteq S \times S$. We write $s \rightarrow s'$ for $(s, s') \in \rightarrow$. Let $\stackrel{*}{\rightarrow}$ be the transitive and reflexive closure of the relation $\rightarrow$. We write $Post_{\mathfrak{S}}(s) = \{s' \in S \mid s \rightarrow s'\}$ for the set of immediate successors of the state s. The *reachability set* of a transition system $\mathfrak{S} = (S, \rightarrow)$ from an initial state s_0 is $Post^*_{\mathfrak{S}}(s_0) = \{s \in S \mid s_0 \stackrel{*}{\rightarrow} s\}$.

We shall be interested in effective transition systems. Intuitively, a transition system $(S, \rightarrow)$ is *effective* iff one can compute the set of successors $Post_{\mathfrak{S}}(s)$ of any state s. We shall take this to imply that $Post_{\mathfrak{S}}(s)$ is finite, and each of its elements is computable.

An *ordered* transition system is a triple $\mathfrak{S} = (S, \rightarrow, \leq)$ where $(S, \rightarrow)$ is a transition system and $\leq$ is a partial ordering on S. We say that $(S, \rightarrow, \leq)$ is *effective* if $(S, \rightarrow)$ is effective and if $\leq$ is decidable.

We say that $\mathfrak{S} = (S, \rightarrow, \leq)$ is *monotonic* (resp. *strictly monotonic*) iff for all $s, s', s_1 \in S$ such that $s \rightarrow s'$ and $s_1 \geq s$ (resp. $s_1 > s$), there exists an $s'_1 \in S$ such that $s_1 \stackrel{*}{\rightarrow} s'_1$ and $s'_1 \geq s'$ (resp. $s'_1 > s'$). $\mathfrak{S}$ is *transitive monotonic* iff for all $s, s', s_1 \in S$ such that $s \rightarrow s'$ and $s_1 \geq s$, there exists an $s'_1 \in S$ such that $s_1 \stackrel{+}{\rightarrow} s'_1$ and $s'_1 \geq s'$. $\mathfrak{S}$ is *strongly monotonic* iff for all $s, s', s_1 \in S$ such that $s \rightarrow s'$ and $s_1 \geq s$, there exists an $s'_1 \in S$ such that $s_1 \rightarrow s'_1$ and $s'_1 \geq s'$. These variations on monotonicity were studied in [Fin87,FS01].

Finite representations of $Post^*_{\mathfrak{S}}(s)$, e.g., as Presburger formulae or finite automata, usually don't exist even for monotonic transition systems (not even speaking of being computable). However, the *cover* $Cover_{\mathfrak{S}}(s) = \downarrow Post^*_{\mathfrak{S}}(\downarrow s)$ $(= \downarrow Post^*_{\mathfrak{S}}(s)$ when $\mathfrak{S}$ is monotonic) will be much better behaved. Note that being able to compute the cover allows one to decide *coverability* ($t \in Cover_{\mathfrak{S}}(s)$?), and *boundedness* (is $Post^*_{\mathfrak{S}}(s)$ finite?). Let us recall that the *control-state reachability problem* (when the set of states is $Q \times X$ with Q a finite set of control states) can be reduced to coverability. However, the *repeated control state reachability problem* (does there exist an infinite computation that visits infinitely often a control state q?) cannot be reduced to coverability.

The *eventuality* property for a given upward closed set I, is the following property: $\mathrm{EG}\, I$ is true in a state s_0 iff there is a computation from s_0 in which all states are in I. Given two labeled transition systems $\mathfrak{S}_1 = (S_1, \rightarrow_1)$ and $\mathfrak{S}_2 = (S_2, \rightarrow_2)$, on the same alphabet Σ, the relation $R \subseteq S_1 \times S_2$ is a *simulation* of $\mathfrak{S}_1$ by $\mathfrak{S}_2$ if for each $(s_1, s_2) \in R$, $s'_1 \in S_1$ and $a \in \Sigma$, if $s_1 \stackrel{a}{\rightarrow} s'_1$ then there exists $s'_2 \in S_2$ such that $s_2 \stackrel{a}{\rightarrow} s'_2$ and $(s'_1, s'_2) \in R$. We say that $s_1 \in S_1$ is

simulated by $s_2 \in S_2$ if there is a simulation R of $\mathfrak{S}_1$ by $\mathfrak{S}_2$ such that $(s_1, s_2) \in R$. An ordered transition system $\mathfrak{S} = (S, \rightarrow, \leq)$ has the *effective PredBasis* property if there exists an algorithm which computes $\uparrow Pre(\uparrow s)$ for each $s \in S$; $\mathfrak{S}$ is *intersection effective* if there is an algorithm which computes a finite basis of $\uparrow s \cap \uparrow s'$, for all states $s, s' \in S$.

Definition 1. *An ordered transition system $\mathfrak{S} = (S, \rightarrow, \leq)$ is a* Well Structured Transition System *(*WSTS*) iff $\mathfrak{S}$ is monotonic and $(S, \leq)$ is wpo. A WSTS $\mathfrak{S} = (S, \rightarrow, \leq)$ is* effective *if $(S, \rightarrow)$ is effective (i.e., $Post(s)$ is finite and computable for any s) and $\leq$ is decidable.*

In particular, an effective WSTS is finitely banching. Some of the decidability results do not require this but, for simplicity, we will make this assumption. Originally, three different definitions of monotonicity (hence six definitions with the strict variant) were given in [Fin87] and four (resp. eight) were studied in [FS01].

We now summarize the main decidability results on WSTS obtained between 1986 and 1996.

Theorem 1. *The following are* decidable*:*

- *Termination, for effective transitive monotonic WSTS [Fin87,FS01].*
- *Boundedness, for effective strictly monotonic transitive WSTS [Fin87,FS01].*
- *Coverability (hence control-state reachability), for effective WSTS with effective PredBasis ([AČJT00], extended in [FS01]).*
- *Eventuality, for effective strongly monotonic finitely branching WSTS (see [KS96,AČJT00], extended in [FS01]).*
- *Simulation of a labeled WSTS by a finite automaton, for intersection effective and effective strongly monotonic WSTS with effective PredBasis [AČJT00].*
- *Simulation of a finite automaton by a labeled WSTS, for effective strongly monotonic WSTS [AČJT00].*

The following are undecidable*:*

- *Reachability, for effective strongly strictly monotonic WSTS (Transfer Petri nets, [DFS98]).*
- *Repeated control-state reachability (hence LTL), for effective strongly strictly monotonic WSTS (Transfer Petri nets, [DFS98]).* □

To prove these decidability results we alternatively use forward and backward algorithms. Termination, boundedness, eventuality and one part of simulation can be proved by using a forward algorithm that builds the so-called Finite Reachability Tree (FRT) [Fin87]: we develop the reachability tree until a state larger than or equal to one of its ancestors is encountered, in which case the current branch is definitely closed. The place-boundedness problem (to decide whether a place can contain an unbounded number of tokens) is undecidable for transfer Petri nets [DFS98], although they are strongly and strictly monotonic WSTS. It is decidable for Petri nets. This requires a richer structure than the

FRT, the Karp-Miller tree. The set of labels of the Karp-Miller tree is a finite representation of the cover.

Almost all the assumptions used above are necessary:

Theorem 2. *The following are* undecidable*:*

- *Termination, for transitive monotonic WSTS.*
- *Boundedness, for effective strongly monotonic WSTS.*
- *Coverability, for effective strongly strictly monotonic WSTS.* □

For termination, Turing machines are transitive monotonic WSTS for which the termination ordering $\leq_{termination}$ is undecidable, [FS01]. For the second claim, Reset Petri nets have an undecidable bounded problem, and are effective strongly monotonic WSTS; but they are not strictly monotonic [DFS98]. For the last claim, there are WSTS composed of two recursive strictly monotonic functions from $\mathbb{N}^2$ into $\mathbb{N}^2$ that are not recursive on $\mathbb{N}^2_\omega$ hence there are no algorithm computing a PredBasis, [FMP04].

In writing this paper, we realized that the status of eventuality and simulation is open: for each of these properties, we know of no natural class of WSTS for which this property would be undecidable.

3.2 WSTS Everywhere: From 1997 to 2012

To the best of our knowledge, there have been no essential new results in the theory of WSTS between 1997 and 2003 (this does not mean that there are no interesting results about *particular* classes of WSTS). Let us just mention two kinds of results: the study of better quasi ordering (bqo) as an alternative to wqo [AN00], and the study of specific models such as Reset/Transfer Petri nets [DFS98], or Lossy Channel Systems [ABJ98]. Moreover, two papers synthesise the known results and show the possible applications: [AČJT00,FS01].

Many papers appeared during the period 2004-2012. We will not make an exhaustive list. Here are some of the papers that introduced new points of view, in our opinion:

2006 P. Ganty, G. Geeraerts, J.-F. Raskin and L. Van Begin proposed [GRvB06a,GRvB06b] a forward procedure for deciding the coverability problem. This is the first forward procedure for this problem in the general framework of WSTS. Their procedure computes a sufficient part (to decide coverability) of the finite representation of the cover.

2007 & 2011 P. Abdulla, G. Delzanno, G. Geeraerts, J.-F. Raskin and L. Van Begin studied [ADB07,GRVB07] the expressive power of WSTS by means of the set of coverability languages which are well-adapted to WSTS. Another, new approach, proposed by R. Bonnet, A. Finkel, S. Haddad and F. Rosa-Velardo in [BFHRV11], is to use the order type of posets to prove, for example, that the class of all WSTS with set of states of type $\mathbb{N}^n$ are less expressive than WSTS with set of states of type $\mathbb{N}^{n+1}$. This strategy unifies the previous proofs and allows us to compare models of different natures, such as lossy channel systems and timed Petri nets.

2004 & 2007 & 2011 R. Lazic, T. Newcomb, J. Ouaknine, A.W. Roscoe, J. Worrell, F. Rosa-Velardo, D. de Frutos-Escrig studied classes of Petri net extensions where tokens carry data: data nets, Petri data nets and ν-Petri nets [LNORW07,VF07,RMF11]. Affine and recursive Petri nets extensions were studied by A. Finkel, P. McKenzie, C. Picaronny in [FMP04]; affine well-structured nets are less expressive than ν-Petri nets.

Since 2009 We began in 2009 a series entitled "Forward analysis for WSTS, Part I: Completions" and "Forward Analysis for WSTS, Part II: Complete WSTS" in which we provide the missing theoretical fundations of finite representations of downward closed sets. This work, based on both order and topology, allowed us to design a new conceptual Karp and Miller procedure. Bounded WSTS [CFS11] are a particular recursive class of WSTS for which the new Karp and Miller procedure terminates.

Since 2010 D. Figueira, S. Figueira, S. Schmitz and Ph. Schnoebelen began the study of the complexity of general WSTS. They characterized the ordinal length of bad sequences of vectors of integers (using the Dickson lemma) and of words (using the Higman lemma) [FFSS11,SS11].

4 Complete WSTS are better

We will now present the recent papers on the computation of a finite representation of the cover. The material of what follows is a part of [FG09b]. We will consider transition systems that are *functional*, i.e., defined by a finite set of transition functions. This is, as in [FG09a], for reasons of simplicity. However, our $\mathbf{Clover}_{\mathfrak{S}}$ procedure (Section 6), and already the technique of *accelerating loops* (Definition 4) depends on the considered transition system being functional. Formally, a *functional transition system* $(S, \xrightarrow{F})$ is a labeled transition system where the transition relation $\xrightarrow{F}$ is defined by a finite set F of partial functions $f : S \longrightarrow S$, in the sense that for every $s, s' \in S$, $s \xrightarrow{F} s'$ iff $s' = f(s)$ for some $f \in F$. If additionally, a partial ordering $\leq$ is given, a map $f : S \to S$ is *partial monotonic* iff $\operatorname{dom} f$ is upward-closed and for all $x, y \in \operatorname{dom} f$ with $x \leq y$, $f(x) \leq f(y)$. An *ordered functional transition system* is an ordered transition system $\mathfrak{S} = (S, \xrightarrow{F}, \leq)$ where F consists of partial monotonic functions. This is always strongly monotonic. A *functional WSTS* is an ordered functional transition system where $\leq$ is a well-ordering.

A functional transition system $(S, \xrightarrow{F})$ is *effective* if every $f \in F$ is computable: given a state s and a function f, we can decide whether $s \in \operatorname{dom} f$ and in this case, one can also compute $f(s)$.

For example, every Petri net, every reset/transfer Petri net, and in fact every affine counter system (see Definition 15) is an effective, functional WSTS.

4.1 Complete WSTS and Their Clovers

All forward procedures for WSTS rest on completing the given WSTS to one that includes all limits. E.g., the state space of Petri nets is $\mathbb{N}^k$, the set of all markings

on k places, but the Karp-Miller algorithm works on $\mathbb{N}_\omega^k$, where $\mathbb{N}_\omega$ is $\mathbb{N}$ plus a new top element ω, with the usual componentwise ordering. We have defined general completions of wpos, serving as state spaces, and have briefly described completions of (functional) WSTS in [FG09a]. We temporarily abstract away from this, and consider *complete* WSTS directly.

Generalizing the notion of continuity to partial maps, we define:

Definition 2. *A* partial continuous *map $f : X \to X$, where $(X, \leq)$ is a dcpo, is a partial map whose domain* $\operatorname{dom} f$ *is open (not just upward-closed), and such that for every directed subset D* in $\operatorname{dom} f$, $\operatorname{lub}(f(D)) = f(\operatorname{lub}(D))$.

This is the special case of a more topological definition: in general, a partial continuous map $f : X \to Y$ is a partial map whose domain is open in X, and such that $f^{-1}(U)$ is open (in X, or equivalently here, in $\operatorname{dom} f$) for any open U of Y.

The composition of two partial continuous maps again yields a partial continuous map.

Definition 3 (Complete WSTS). *A* complete *transition system is a functional transition system $\mathfrak{S} = (S, \xrightarrow{F}, \leq)$ where $(S, \leq)$ is a continuous dcwo and every function in F is partial continuous. A* complete WSTS *is a functional WSTS that is complete as a functional transition system.*

The point in complete WSTS is that one can *accelerate* loops:

Definition 4 (Lub-acceleration). *Let $(X, \leq)$ be a dcpo, $f : X \to X$ be partial continuous. The* lub-acceleration *$f^\infty : X \to X$ is defined by:* $\operatorname{dom} f^\infty = \operatorname{dom} f$, *and for any $x \in \operatorname{dom} f$, if $x < f(x)$ then $f^\infty(x) = \operatorname{lub}\{f^n(x) \mid n \in \mathbb{N}\}$, else $f^\infty(x) = f(x)$.*

Note that if $x \leq f(x)$, then $f(x) \in \operatorname{dom} f$, and $f(x) \leq f^2(x)$. By induction, we can show that $\{f^n(x) \mid n \in \mathbb{N}\}$ is an increasing sequence, so that the definition makes sense.

Complete WSTS are strongly monotonic. One cannot decide, in general, whether a recursive function f is monotonic [FMP04] or continuous, whether an ordered set $(S, \leq)$ with a decidable ordering $\leq$, is a dcpo or whether it is a wpo. To show the latter claim for example, fix a finite alphabet Σ, and consider subsets S of Σ^* specified by a Turing machine $\mathcal{M}$ with tape alphabet Σ, so that S is the language accepted by $\mathcal{M}$.

We can also prove that given an effective ordered functional transition system, one cannot decide whether it is a WSTS, or a complete WSTS, in a similar way. However, the completion of *any* functional ω^2-WSTS is complete, as we shall see in Theorem 3.

In a complete WSTS, there is a *canonical* finite representation of the cover: the *clover* (a succinct description of the *cl*osure of the c*over*).

Definition 5 (Clover). *Let $\mathfrak{S} = (S, \xrightarrow{F}, \leq)$ be a complete WSTS. The* clover *$Clover_{\mathfrak{S}}(s_0)$ of the state $s_0 \in S$ is* $\operatorname{Max}\operatorname{Lub}(Cover_{\mathfrak{S}}(s_0))$.

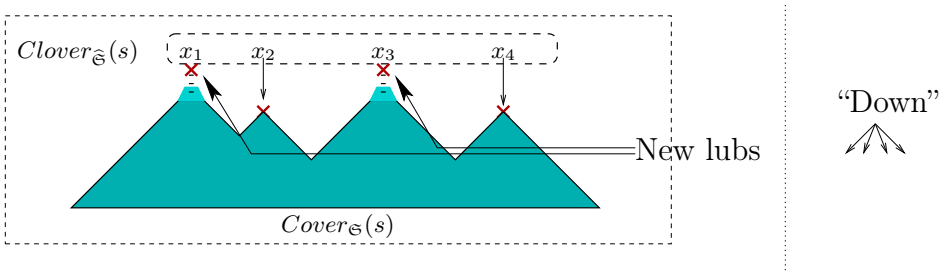

Fig. 1. The clover and the cover, in a complete space

This is illustrated in Figure 1. The "down" part on the right is meant to illustrate in which directions one should travel to go down in the chosen ordering. The cover $Cover_{\mathfrak{S}}(s_0)$ is a downward-closed subset, illustrated in blue (grey if you read this in black and white). $\mathrm{Lub}(Cover_{\mathfrak{S}}(s_0))$ has some new least upper bounds of directed subsets, here x_1 and x_3. The clover is given by just the maximal points in $\mathrm{Lub}(Cover_{\mathfrak{S}}(s_0))$, here x_1, x_2, x_3, x_4.

The fact that the clover is indeed a representation of the cover follows from the following.

Lemma 1. *Let $(S, \leq)$ be a continuous dcwo. For any closed subset F of S, $\mathrm{Max}\, F$ is finite and $F = {\downarrow}\mathrm{Max}\, F$.*

Proposition 1. *Let $\mathfrak{S} = (S, \xrightarrow{F}, \leq)$ be a complete WSTS, and $s_0 \in S$. Then $Clover_{\mathfrak{S}}(s_0)$ is finite, and $cl(Cover_{\mathfrak{S}}(s_0)) = {\downarrow}Clover_{\mathfrak{S}}(s_0)$.*

For any other representative, i.e., for any finite set R such that ${\downarrow}R = {\downarrow}Clover_{\mathfrak{S}}(s_0)$, $Clover_{\mathfrak{S}}(s_0) = \mathrm{Max}\, R$. Indeed, for any two finite sets $A, B \subseteq S$ such that ${\downarrow}A = {\downarrow}B$, $\mathrm{Max}\, A = \mathrm{Max}\, B$. So *Clover* is the *minimal representative* of the cover, i.e., there is no representative R with $|R| < |Clover_{\mathfrak{S}}(s_0)|$. The clover was called the minimal coverability set in [Fin93].

Despite the fact that the clover is always finite, it is non-computable in general (for example for Reset Petri nets) Nonetheless, it is computable on *flat* complete WSTS, and even on the larger class of *clover-flattable* complete WSTS (Theorem 7 below).

4.2 Completions

Many WSTS are not complete: the set $\mathbb{N}^k$ of states of a Petri net with k places is not even a dcpo. The set of states of a lossy channel system with k channels, $(\Sigma^*)^k$, is not a dcpo for the subword ordering either. We have defined general completions of wpos, and of WSTS, in [FG09a], a construction which we recall quickly.

The *completion* $\widehat{X}$ of a wpo $(X, \leq)$ is defined in any of two equivalent ways. First, $\widehat{X}$ is the *ideal completion* $\mathrm{Idl}(X)$ of X, i.e., the set of ideals of X, ordered

by inclusion, where an *ideal* is a downward-closed directed subset of X. The least upper bound of a directed family of ideals $(D_i)_{i\in I}$ is their union. $\widehat{X}$ can also be described as the sobrification $\mathcal{S}(X_a)$ of the Noetherian space X_a, but this is probably harder to understand.

There is an embedding $\eta_X : X \to \widehat{X}$, i.e., an injective map such that $x \leq x'$ in X iff $\eta_X(x) \leq \eta_X(x')$ in $\widehat{X}$. This is defined by $\eta_X(x) = \downarrow x$. This allows us to consider X as a subset of $\widehat{X}$, by equating X with its image $\eta_X\langle X\rangle$, i.e., by equating each element $x \in X$ with $\downarrow x \in \widehat{X}$. However, we shall only do this in informal discussions, as this tends to make proofs messier.

For instance, if $X = \mathbb{N}^k$, e.g., with $k = 3$, then $(1, 3, 2)$ is equated with the ideal $\downarrow(1, 3, 2)$, while $\{(1, m, n) \mid m, n \in \mathbb{N}\}$ is a *limit*, i.e. an element of $\widehat{X} \setminus X$; the latter is usually written $(1, \omega, \omega)$, and is the least upper bound of all $(1, m, n)$, $m, n \in \mathbb{N}$. The downward-closure of $(1, \omega, \omega)$ in $\widehat{X}$, intersected with X, gives back the set of non-limit elements $\{(1, m, n) \mid m, n \in \mathbb{N}\}$.

This is a general situation: one can always write $\widehat{X}$ as the disjoint union $X \cup L$, so that any downward-closed subset D of X can be written as $X \cap \downarrow A$, where A is a *finite* subset of $X \cup L$. Then L, the set of limits, is a *weak adequate domain of limits* (WADL) for X—we slightly simplify Definition 3.1 of [FG09a], itself a slight generalization of [GRvB06b]. In fact, $\widehat{X}$ (minus X) is the *smallest* WADL [FG09a, Theorem 3.4].

$\widehat{X} = \mathrm{Idl}(X)$ is always a continuous dcpo. In fact, it is even algebraic [AJ94, Proposition 2.2.22]. It may however fail to be well, hence to be a continuous dcwo, see [FG12b, Section 4.2].

We have also described a hierarchy of datatypes on which completions are effective [FG09a, Section 5]. Notably, $\widehat{\mathbb{N}} = \mathbb{N}_\omega$, $\widehat{A} = A$ for any finite poset, and $\widehat{\prod_{i=1}^k X_i} = \prod_{i=1}^k \widehat{X}_i$. Also, $\widehat{X^*}$ is the space of *word-products* on X. These are the products, as defined in [ABJ98], i.e., regular expressions that are products of *atomic expressions* A^* ($A \in \mathbb{P}_{\mathrm{fin}}(\widehat{X})$, $A \neq \emptyset$) or $a^?$ ($a \in \widehat{X}$). In any case, elements of completions $\widehat{X}$ have a finite description, and the ordering $\subseteq$ on elements of $\widehat{X}$ is decidable [FG09a, Theorem 5.3].

Having defined the completion $\widehat{X}$ of a wpo X, we can define the completion $\mathfrak{S} = \widehat{\mathfrak{X}}$ of a (functional) WSTS $\mathfrak{X} = (X, \xrightarrow{F}, \leq)$ as $(\widehat{X}, \xrightarrow{\mathcal{S}F}, \subseteq)$, where $\mathcal{S}F = \{\mathcal{S}f \mid f \in F\}$ [FG09a, Section 6]. For each partial monotonic map $f \in F$, the partial continuous map $\mathcal{S}f : \widehat{X} \to \widehat{X}$ is such that $\mathrm{dom}\,\mathcal{S}f = \{C \in \widehat{X} \mid C \cap \mathrm{dom}\, f \neq \emptyset\}$, and $\mathcal{S}f(C) = \downarrow f\langle C\rangle$ for every $C \in \widehat{X}$. In the cases of Petri nets or functional-lossy channel systems, the completed WSTS is effective [FG09a, Section 6].

The important fact, which assesses the importance of the clover, is Proposition 2 below. We first require a useful lemma. Up to the identification of X with its image $\eta_X\langle X\rangle$, this states that for any downward-closed subset F of $\widehat{X}$, $cl(F) \cap X = F \cap X$, i.e., taking the closure of F only adds new limits, no proper elements of X.

Up to the identification of X with $\eta_X\langle X\rangle$, the next proposition states that $Cover_{\mathfrak{X}}(s_0) = Cover_{\mathfrak{S}}(s_0) \cap X = \downarrow Clover_{\mathfrak{S}}(s_0) \cap X$. In other words, to compute the cover of s_0 in the WSTS $\mathfrak{X}$ on the state space X, one can equivalently

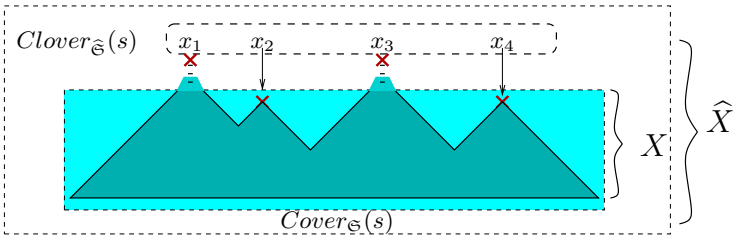

Fig. 2. The clover and the cover, in a completed space

compute the cover s_0 in the completed WSTS $\widehat{\mathfrak{X}}$, and keep only those non-limit elements (first equality of Proposition 2). Or one can equivalently compute the *closure* of the cover in the completed WSTS $\widehat{\mathfrak{X}}$, in the form of the downward closure $\downarrow Clover_{\mathfrak{S}}(s_0)$ of its clover. The closure of the cover will include extra limit elements, compared to the cover, but no non-limit element. This is illustrated in Figure 2.

Proposition 2. *Let $\mathfrak{S} = \widehat{\mathfrak{X}}$ be the completion of the functional WSTS $\mathfrak{X} = (X, \xrightarrow{F}, \leq)$. For every state $s_0 \in X$, $Cover_{\mathfrak{X}}(s_0) = \eta_X^{-1}(Cover_{\mathfrak{S}}(\eta_X(s_0))) = \eta_X^{-1}(\downarrow Clover_{\mathfrak{S}}(\eta_X(s_0)))$.*

$Cover_{\mathfrak{S}}(s_0)$ is contained, usually strictly, in $\downarrow Clover_{\mathfrak{S}}(s_0)$. The above states that, when restricted to non-limit elements (in X), both contain the same elements. Taking lub-accelerations $(\mathcal{S}f)^\infty$ of any composition f of maps in F may leave $Cover_{\mathfrak{S}}(s_0)$, but is always contained in $\downarrow Clover_{\mathfrak{S}}(s_0) = cl(Cover_{\mathfrak{S}}(s_0))$. So we can safely lub-accelerate in $\mathfrak{S} = \widehat{\mathfrak{X}}$ to compute the clover in $\mathfrak{S}$. While the clover is larger than the cover, taking the intersection back with X will produce exactly the cover $Cover_{\mathfrak{X}}(s_0)$.

In more informal terms, the cover is the set of states reachable by either following the transitions in F, or going down. The closure of the cover $\downarrow Clover_{\mathfrak{S}}(s_0)$ contains not just states that are reachable in the above sense, but also the limits of chains of such states. One may think of the elements of $\downarrow Clover_{\mathfrak{S}}(s_0)$ as being those states that are "reachable in infinitely many steps" from s_0. And we hope to find the finitely many elements of $Clover_{\mathfrak{S}}(s_0)$ by doing enough lub-accelerations.

5 Completion of WSTS into complete WSTS is (almost) always possible

It would seem clear that the construction of the completion $\mathfrak{S} = \widehat{\mathfrak{X}}$ of a WSTS $\mathfrak{X} = (X, \xrightarrow{F}, \leq)$ be, again, a WSTS. We shall show that this is not the case. The only missing ingredient to show that $\mathfrak{S}$ is a complete WSTS is to check that $\widehat{X}$ is well-ordered by inclusion. We have indeed seen that $\widehat{X}$ is a continuous dcpo;

and $\mathfrak{S}$ is strongly monotonic, because $\mathcal{S}f$ is continuous, hence monotonic, for every $f \in F$.

Next, we shall concern ourselves with the question: under what condition on $\mathfrak{X}$ is $\mathfrak{S} = \widehat{\mathfrak{X}}$ again a WSTS? Equivalently, when is $\widehat{X}$ well-ordered by inclusion? We shall see that there is a definite answer: when X is ω^2-wqo.

5.1 Motivation

The question may seem mostly of academic interest. Instead, we illustrate that it is crucial to establish a *progress* property described below.

Let us imagine a procedure in the style of the Karp-Miller tree construction. We shall provide an abstract version of one, $\mathbf{Clover}_{\mathfrak{S}}$, in Section 6. However, to make things clearer, we shall use a direct imitation of the Karp-Miller procedure for Petri nets for now, generalized to arbitrary WSTS. This is a slight variant of the *generalized Karp-Miller procedure* of [Fin87,Fin90], and we shall therefore call it as such.

We build a tree, with nodes labeled by elements of the completion $\widehat{X}$, and edges labelled by transitions $f \in F$. During the procedure, nodes can be marked extensible or non-extensible. We start with the tree with only one node labeled s_0, and mark it extensible. At each step of the procedure, we pick an extensible leaf node N, labeled with $s \in \widehat{X}$, say, and add new children to N. For each $f \in F$ such that $s \in \operatorname{dom} \mathcal{S}f$, let $s' = \mathcal{S}f(s)$, and add a new child N' to N. The edge from N to N' is labeled f. If s' already labels some ancestor of N', then we label N' with s' and mark it non-extensible. If $s'' \leq s'$ for no label s'' of an ancestor of N', then we label N' with s' and mark it extensible. Finally, if $s'' < s'$ for some label s'' of an ancestor N_0 of N' (what we shall refer to as case (*) below), then the path from N_0 to N' is labeled with a sequence of functions $f_1, \ldots, f_p$ from F, and we label N' with the lub-acceleration $(f_p \circ \ldots \circ f_1)^\infty(s'')$. (There is a subtle issue here: if there are several such ancestors N_0, then we possibly have to lub-accelerate several sequences $f_1, \ldots, f_p$ from the label s'' of N_0: in this case, we must create several successor nodes N', one for each value of $(f_p \circ \ldots \circ f_1)^\infty(s'')$.) When $X = \mathbb{N}^k$ and each $f \in F$ is a Petri net transition, this is the Karp-Miller procedure, up to the subtle issue just mentioned, which we shall ignore.

Let us recall that the Karp-Miller tree (and also the reachability tree) is *finitely branching*, since the set F of functions is finite. This will allow us to use König's Lemma, which states that any finitely branching, infinite tree has at least one infinite branch.

The reasons why the original Karp-Miller procedure terminates on (ordinary) Petri nets are two-fold. First, when $\widehat{X} = \mathbb{N}^k_\omega$, one cannot lub-accelerate more than k times, because each lub-acceleration introduces a new ω component to the label of the produced state, which will not disappear in later node extensions. This is specific to Petri nets, and already fails for reset Petri nets, where ω components do disappear.

The second reason is of more general applicability: $\widehat{X} = \mathbb{N}^k_\omega$ is wpo, and this implies that along every infinite branch of the tree thus constructed, case (*)

will eventually happen, and in fact will happen infinitely many times. Call this *progress*: along any infinite path, one will lub-accelerate infinitely often. In the original Karp-Miller procedure for Petri nets, this will entail termination.

As we have already announced, for WSTS other than Petri nets, termination cannot be ensured. But at least we would like to ensure progress. The argument above shows that progress is obtained provided $\widehat{X}$ is wpo (or even just wqo). *This* is our main motivation in characterizing those wpos X such that $\widehat{X}$ is wpo again.

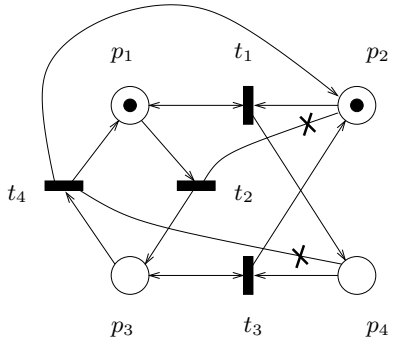

Fig. 3. The reset Petri net from [DFS98]

Before we proceed, let us explain why termination cannot be ensured. Generally, this will follow from undecidability arguments Here is a concrete case of non-termination. Consider the reset Petri net of [DFS98, Example 3], see Figure 3. This net has 4 places and 4 transitions, hence defines an transition system on $\mathbb{N}^4$. Its transitions are: $t_1(n_1, n_2, n_3, n_4) = (n_1, n_2 - 1, n_3, n_4 + 1)$ if $n_1, n_2 \geq 1$, $t_2(n_1, n_2, n_3, n_4) = (n_1 - 1, 0, n_3 + 1, n_4)$ if $n_1 \geq 1$, $t_3(n_1, n_2, n_3, n_4) = (n_1, n_2 + 1, n_3, n_4 - 1)$ if $n_3, n_4 \geq 1$, and $t_4(n_1, n_2, n_3, n_4) = (n_1 + 1, n_2 + 1, n_3 - 1, 0)$ if $n_3 \geq 1$. Note that $t_4(t_3^{n_2}(t_2(t_1^{n_2}(1, n_2, 0, 0)))) = (1, n_2 + 1, 0, 0)$ whenever $n_2 \geq 1$. The generalized Karp-Miller tree procedure, starting from $s_0 = (1, 1, 0, 0)$, will produce a child labeled $(1, 0, 0, 1)$ through t_1, then $(0, 0, 1, 1)$ through t_2, then $(0, 1, 1, 0)$ through t_3. Using t_4 leads us to case (*) with $s' = (1, 2, 0, 0)$. So the procedure will lub-accelerate the sequence $t_1 t_2 t_3 t_4$, starting from $s_0 = (1, 1, 0, 0)$. However $(t_4 \circ t_3 \circ t_2 \circ t_1)(s') = (1, 1, 0, 0) = s'$ again, so the sequence of iterates $(t_4 \circ t_3 \circ t_2 \circ t_1)^n(s_0)$ stabilizes at s', and $(t_4 \circ t_3 \circ t_2 \circ t_1)^\infty(s_0) = s'$. So the procedure adds a node labeled $s' = (1, 2, 0, 0)$. Similarly, starting from the latter, the procedure will eventually lub-accelerate the sequence $t_1^2 t_2 t_3^2 t_4$, producing a node labeled $(1, 3, 0, 0)$, and in general produce nodes labeled $(1, i + 1, 0, 0)$ for any $i \geq 1$ after having lub-accelerated the sequence $t_1^i t_2 t_3^i t_4$ from a node labeled $(1, i, 0, 0)$. In particular, the generalized Karp-Miller tree procedure will generate infinitely many nodes, and therefore fail to terminate.

This example also illustrates the following: progress does *not* mean that we shall eventually compute limits $g^\infty(s)$ that could not be reached in finitely many steps. In the example above, we do lub-accelerate infinitely often, and compute $(t_4 \circ t_3^i \circ t_2 \circ t_1^i)^\infty(1,i,0,0)$, but none of these lub-accelerations actually serve any purpose, since $(t_4 \circ t_3^i \circ t_2 \circ t_1^i)^\infty(1,i,0,0) = (1,i+1,0,0)$ is already equal to $(t_4 \circ t_3^i \circ t_2 \circ t_1^i)(1,i,0,0)$.

Progress will take a slightly different form in the actual procedure $\mathbf{Clover}_{\mathfrak{S}}$ of Section 6. In fact, the latter will not build a tree, as the tree is in fact only algorithmic support for ensuring a fair choice of a state in $\widehat{X}$, and essentially acts as a distraction. However, progress will be crucial (Proposition 5 states that if the set of values computed by the procedure $\mathbf{Clover}_{\mathfrak{S}}$ is finite then $\mathbf{Clover}_{\mathfrak{S}}$ terminates) in our characterization of the cases where $\mathbf{Clover}_{\mathfrak{S}}$ terminates (Theorem 7), as those states that are clover-flattable (see Section 6). Without it, $\mathbf{Clover}_{\mathfrak{S}}$ would terminate in strictly less cases.

5.2 ω^2-WSTS

Recall here the working definition in [Jan99]: a well-quasi-order X is *ω^2-wqo* if and only if $(\mathbb{P}(X), \leq^\sharp)$ is wqo (where $A \leq^\sharp B$ iff for every $b \in B$, there is an $a \in A$ such that $a \leq b$ or equivalently iff $\uparrow B \subseteq \uparrow A$ iff $B \subseteq \uparrow A$). We show that the above is the only case that can go bad:

Proposition 3. *Let S be a well-quasi-order. Then $\widehat{S}$ is well-quasi-ordered by inclusion iff S is ω^2-wqo.*

Let an *ω^2-WSTS* be any WSTS whose underlying poset is ω^2-wqo. It follows:

Theorem 3. *Let $\mathfrak{S} = (S, \xrightarrow{F}, \leq)$ be a functional WSTS. Then $\widehat{\mathfrak{S}}$ is a (complete, functional) WSTS iff $\mathfrak{S}$ is an ω^2-WSTS.* □

5.3 Are ω^2-wqos Ubiquitous?

It is natural to ask whether this is the norm or an exception. We claim that all wpos used in the verification literature are in fact ω^2-wpo.

Consider the following grammar of datatypes, which extends that of [FG09a, Section 5] with the case of finite trees (last line):

$$\begin{array}{lll} D ::= \mathbb{N} & & \text{natural numbers} \\ \quad | \quad A_\leq & & \text{finite set } A\text{, ordered by } \leq \\ \quad | \quad D_1 \times \ldots \times D_k & & \text{finite product} \\ \quad | \quad D_1 + \ldots + D_k & & \text{finite, disjoint sum} \\ \quad | \quad D^* & & \text{finite words} \\ \quad | \quad D^{\circledast} & & \text{finite multisets} \\ \quad | \quad \mathcal{T}(D) & & \text{finite trees} \end{array} \tag{1}$$

Then:

Proposition 4. *Every datatype defined in (1) is ω^2-wqo, and in fact bqo.*

In fact, all naturally occurring wqos are bqos, perhaps to the notable exception of finite graphs quasi-ordered by the graph minor relation, which are wqo [RS04] but not known to be bqo.

5.4 Effective Complete WSTS

The completion $\widehat{\mathfrak{S}}$ of a WSTS $\mathfrak{S}$ is effective iff the completion $\widehat{S}$ of the set of states is effective and $\mathcal{S}f$ is recursive for all $f \in F$. $\widehat{S}$ is effective for all the data types of [FG09a, Section 5]

Also, $\mathcal{S}f$ is indeed recursive for all $f \in F$, whether in Petri nets, functional-lossy channel systems, and reset/transfer Petri nets notably.

In the case of ordinary or reset/transfer Petri nets, and in general for all affine counter systems (which we shall investigate from Definition 15 on), $\mathcal{S}f$ coincides with the extension $\overline{f}$ defined in [FMP04, Section 2]: whenever $\operatorname{dom} f$ is upward-closed and $f : \mathbb{N}^k \to \mathbb{N}^k$ is defined by $f(\boldsymbol{s}) = A\boldsymbol{s} + \boldsymbol{a}$, for some matrix $A \in \mathbb{N}^{k \times k}$ and vector $\boldsymbol{a} \in \mathbb{Z}^k$, then $\operatorname{dom} \mathcal{S}f = \uparrow_S \operatorname{dom} f$, and $\mathcal{S}(f)(\boldsymbol{s})$ is again defined as $A\boldsymbol{s} + \boldsymbol{a}$, this time for all $\boldsymbol{s} \in \mathbb{N}_\omega^k$, and using the convention that $0 \times \omega = 0$ when computing the matrix product $A\boldsymbol{s}$ [FMP04, Theorem 7.9].

6 A Conceptual Karp-Miller Procedure

There are some advantages in using a forward procedure to compute (part of) the clover for solving coverability. For depth-bounded processes, a fragment of the π-calculus, the simple algorithm that works backward (computing the set of predecessors of an upward-closed initial set) of [AČJT00] is not applicable when the maximal depth of configurations is not known in advance because, in this case, the predecessor configurations are not effectively computable [WZH10]. It has been also proved that, unlike backward algorithms (which solve coverability without computing the clover), the Expand, Enlarge and Check forward algorithm of [GRvB07], which operates on complete WSTS, solves coverability by computing a *sufficient* part of the clover, even though the depth of the process is not known a priori [WZH10]. Recently, Zufferey, Wies and Henzinger proposed to compute a part of the clover by using a particular widening, called a *set-widening operator* [ZWH12], which loses some information, but always terminates and seems sufficiently precise to compute the clover in various case studies.

Model-checking safety properties of WSTS can be reduced to coverability, but there are other properties, such as *boundedness* (is $Post^*_{\mathfrak{S}}(s)$ finite?) that cannot be reduced to coverability: boundedness is decidable for Petri nets but undecidable for Reset Petri nets [DFS98], hence for general WSTS.

Recall that being able to compute the clover allows one to decide not only *coverability* since t is coverable from s iff $t \in Cover_{\mathfrak{S}}(s)$ iff $\exists t' \in Clover_{\mathfrak{S}}(s)$ such that $t \leq t'$ but also boundedness, and place-boundedness. To the best of our knowledge, the only known algorithms that decide place-boundedness (and

also some formal language properties such as regularity and context-freeness of Petri net languages) *require one to compute the clover*.

Another argument in favor of computing clovers is Emerson and Namjoshi's [EN98] approach to model-checking *liveness* properties of WSTS, which uses a finite (coverability) graph based on the clover. Since WSTS enjoy the finite path property ([EN98], Definition 7), model-checking liveness properties is decidable for complete WSTS for which the clover is computable.

All these reasons motivate us to *try* to compute the clover for classes of complete WSTS, even though it is not computable in general.

The key to designing some form of a Karp-Miller procedure, such as the generalized Karp-Miller tree procedure (Section 5.1) or the $\mathbf{Clover}_{\mathfrak{S}}$ procedure below is being able to *compute* lub-accelerations. Hence:

Definition 6 (∞-Effective). *An effective complete functional WSTS $\mathfrak{S} = (S, \xrightarrow{F}, \leq)$ is* ∞-effective *iff every function g^∞ is computable, for every $g \in F^*$, where F^* is the set of all compositions of maps in F.*

E.g., the completion of a Petri net is ∞-effective: not only is $\mathbb{N}^k_\omega$ a wpo, but every composition of transitions $g \in F^*$ is of the form $g(\boldsymbol{x}) = \boldsymbol{x} + \delta$, where $\delta \in \mathbb{Z}^k$. If $\boldsymbol{x} < g(\boldsymbol{x})$ then $\delta \in \mathbb{N}^k \setminus \{0\}$. Write $\boldsymbol{x}_i$ the ith component of $\boldsymbol{x}$, it follows that $g^\infty(\boldsymbol{x})$ is the tuple whose ith component is $\boldsymbol{x}_i$ if $\delta_i = 0$, ω otherwise.

Let $\mathfrak{S}$ be an ∞-effective WSTS, and write $A \leq^\flat B$ iff $\downarrow A \subseteq \downarrow B$, i.e., iff every element of A is below some element of B. This is the *Hoare quasi-ordering*, also known as the *domination* quasi-ordering. The following is a simple procedure which computes the clover of its input $s_0 \in S$ (when it terminates):

Procedure $\mathbf{Clover}_{\mathfrak{S}}(s_0)$:
1. $A \leftarrow \{s_0\}$;
2. **while** $Post_{\mathfrak{S}}(A) \not\leq^\flat A$ **do**
   (a) Choose fairly (see below) $(g, a) \in F^* \times A$ such that $a \in \operatorname{dom} g$;
   (b) $A \leftarrow A \cup \{g^\infty(a)\}$;
3. **return** $\operatorname{Max} A$;

Note that $\mathbf{Clover}_{\mathfrak{S}}$ is well-defined and all its lines are computable by assumption, provided we make clear what we mean by fair choice in line (a). Call A_m the value of A at the start of the $(m-1)$st turn of the loop at step 2 (so in particular $A_0 = \{s_0\}$). The choice at line (a) is *fair* iff, on every infinite execution, every pair $(g, a) \in F^* \times A_m$ will be picked at some later stage $n \geq m$.

A possible implementation of this fair choice is the generalized Karp-Miller tree construction of Section 5.1: organize the states of A as labeling nodes of a tree that we grow. At step m, A_m is the set of leaves of the tree, and case (*) of the generalized Karp-Miller tree construction ensures that all pairs $(g, a) \in F^* \times A_m$ will eventually be picked for consideration. However, the generalized Karp-Miller tree construction does some useless work, e.g., when two nodes of the tree bear the same label.

Most existing proposals for generalizing the Karp-Miller construction do build such a tree [KM69,Fin90,Fin93,GRvB07], or a graph [EN98]. We claim

that this is mere algorithmic support for ensuring fairness, and that the goal of such procedures is to compute a finite representation of the cover. Our $\mathbf{Clover}_{\mathfrak{S}}$ procedure computes the clover, which is the minimal such representation, and isolates algorithmic details from the core construction.

We shall also see that termination of $\mathbf{Clover}_{\mathfrak{S}}$ has strong ties with the theory of *flattening* [BFLS05]. However, Bardin *et al.* require one to enumerate sets of the form $g^*(\boldsymbol{x})$, which is sometimes harder than computing the single element $g^\infty(\boldsymbol{x})$. For example, if $g : \mathbb{N}^k \to \mathbb{N}^k$ is an affine map $g(\boldsymbol{x}) = A\boldsymbol{x} + \boldsymbol{b} - \boldsymbol{a}$ for some matrix $A \in \mathbb{N}^{k\times k}$ and vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{N}^k$, then $g^\infty(\boldsymbol{x})$ is computable as a vector in $\mathbb{N}^k_\omega$, as we have seen in Section 5.4. But $g^*(\boldsymbol{x})$ is not even definable by a Presburger formula in general, in fact even when g is a composition of Petri net transitions; this is because reachability sets of Petri nets are not semi-linear in general [HP79].

Finally, we use a *fixpoint test* (line 2) that is not in the Karp-Miller algorithm; and this improvement allows $\mathbf{Clover}_{\mathfrak{S}}$ to terminate in *more cases* than the Karp-Miller procedure when it is used for extended Petri nets (for reset Petri nets for instance, which are a special case of the affine maps above), as we shall see. To decide whether the current set A, which is always an under-approximation of $Clover_{\mathfrak{S}}(s_0)$, is the clover, it is enough to decide whether $Post_{\mathfrak{S}}(A) \leq^\flat A$. The various Karp-Miller procedures only test each branch of a tree separately, to the partial exception of the minimal coverability tree algorithm [Fin90] and Geeraerts *et al.*'s recent coverability algorithm [GRvB07], which compare nodes across branches. That the simple test $Post_{\mathfrak{S}}(A) \leq^\flat A$ does all this at once does not seem to have been observed until now.

6.1 Correctness and Termination of the Clover Procedure

We cannot hope to have $\mathbf{Clover}_{\mathfrak{S}}$ terminate on all inputs. But we can at least start by showing that it is correct, whenever it terminates. This will be Theorem 4 below.

We first show that if $\mathbf{Clover}_{\mathfrak{S}}$ terminates then the computed set A is contained in $\mathrm{Lub}(Post^*_{\mathfrak{S}}(s_0))$. It is crucial that $\mathrm{Lub}(F) = cl(F)$ for any downward-closed set F, which holds because the state space S is a continuous dcpo. We use this through invocations to Proposition 1.

If the procedure $\mathbf{Clover}_{\mathfrak{S}}$ does not stop, it will compute an infinite sequence of sets of states. In other words, $\mathbf{Clover}_{\mathfrak{S}}$ does not deadlock. This is the progress property mentioned in Section 5.1.

Proposition 5 (Progress). *Let $\mathfrak{S}$ be an ∞-effective complete functional WSTS and A_n be the value of the set A, computed by the procedure $\mathbf{Clover}_{\mathfrak{S}}$ on input s_0, after n iterations of the while statement at line 2. If $\bigcup_n A_n$ is finite, then the procedure $\mathbf{Clover}_{\mathfrak{S}}$ terminates on input s_0.*

While $\mathbf{Clover}_{\mathfrak{S}}$ is non-deterministic, this is *don't care non-determinism*: if one execution does not terminate, then no execution terminates. If $\mathbf{Clover}_{\mathfrak{S}}$ terminates, then it computes the clover, and if it does not terminate, then at

each step n, the set A_n is contained in the clover. Let us recall that $A_n \leq^\flat A_{n+1}$. We can now prove:

Theorem 4 (Correctness). *If* $\mathbf{Clover}_{\mathfrak{S}}(s_0)$ *terminates, then it computes* $Clover_{\mathfrak{S}}(s_0)$.

If the generalized Karp-Miller tree procedure (see Section 5.1) terminates then it has found a finite set $g_1, g_2, ..., g_n$ of maps to lub-accelerate. These lub-accelerations will also be found by $\mathbf{Clover}_{\mathfrak{S}}$, by fairness. From the fixpoint test, $\mathbf{Clover}_{\mathfrak{S}}$ will also stop. So $\mathbf{Clover}_{\mathfrak{S}}$ terminates on at least all inputs where the generalized Karp-Miller tree procedure terminates. We can say more:

Proposition 6. *The procedure* $\mathbf{Clover}_{\mathfrak{S}}$ *terminates on strictly more input states* $s_0 \in S$ *than the generalized Karp-Miller tree procedure.*

Proof. Consider the reset Petri net of [DFS98, Example 3] again (Figure 3). Add a new transition $t_5(n_1, n_2, n_3, n_4) = (n_1 + 1, n_2 + 1, n_3 + 1, n_4 + 1)$. The generalized Karp-Miller procedure does not terminate on this modified reset Petri net starting from $s_0 = (1, 1, 0, 0)$, because it already does not terminate on the smaller one of Section 5.1. On the other hand, by fairness, $\mathbf{Clover}_{\mathfrak{S}}$ will sooner or later decide to pick a pair of the form (t_5, a) at line (a), and then immediately terminate with the maximal state $(\omega, \omega, \omega, \omega)$, which is the sole element of the clover. □

Deciding when $\mathbf{Clover}_{\mathfrak{S}}$ terminates is itself impossible. We first observe that $\mathbf{Clover}_{\mathfrak{S}}$ terminates on each bounded state.

Lemma 2. *Let* $\mathfrak{S} = (S, \xrightarrow{F})$ *be an* ∞*-effective complete WSTS, and* $s_0 \in S$ *a state such that the reachability set* $Post^*_{\mathfrak{S}}(s_0)$ *is finite. Then* $\mathbf{Clover}_{\mathfrak{S}}(s_0)$ *terminates.*

Proof. Since $Post^*_{\mathfrak{S}}(s_0)$ is finite, $g^\infty(s)$ is in $Post^*_{\mathfrak{S}}(s_0)$ for every $s \in Post^*_{\mathfrak{S}}(s_0)$ and every $g \in F^*$ with $s \in \operatorname{dom} g$. So, defining again A_n as the value of the set A computed by $\mathbf{Clover}_{\mathfrak{S}}$ on input s_0, after n iterations of the while statement at line 2, $\bigcup_{n\in\mathbb{N}} A_n$ is contained in $Post^*_{\mathfrak{S}}(s_0)$, hence finite. By Proposition 5, $\mathbf{Clover}_{\mathfrak{S}}(s_0)$ terminates. □

Proposition 7. *There is an* ∞*-effective complete WSTS* $\mathfrak{S} = (S, \xrightarrow{F})$ *such that we cannot decide, given* $s_0 \in S$*, whether* $\mathbf{Clover}_{\mathfrak{S}}(s_0)$ *will terminate.*

The following result was more generally stated in [Fin87] (but without sufficient effective and completeness hypotheses) and it was also expressed for Recursive Well Structured Nets in [FMP04] where the ∞-effective hypothesis was replaced by a weaker condition that allows to compute a sufficient underapproximation of the limit of the $f^n(x)$ when n goes to infinity and for $x < f(x)$.

Theorem 5. *[BF12] For* ∞*-effective strictly monotonic complete WSTS* $\mathfrak{S} = (\mathbb{N}^n, \xrightarrow{F}, \leq)$*, the procedure* $\mathbf{Clover}_{\mathfrak{S}}(s_0)$ *terminates.*

There is another case in which the procedure $\mathbf{Clover}_{\mathfrak{S}}$ terminates. A functional transition system $\mathfrak{S} = (S, \xrightarrow{F})$ with initial state s_0 is *flat* iff there are finitely many words $w_1, w_2, ..., w_k \in F^*$ such that any fireable sequence of transitions from s_0 is contained in the language $w_1^* w_2^* ... w_k^*$. (We equate functions in F with letters from the alphabet F.) corresponding composition of maps, i.e., fg denotes $g \circ f$.) Ginsburg and Spanier [GS64] call this a *bounded* language, and show that it is decidable whether any context-free language is flat.

Theorem 6. *For ∞-effective complete flat WSTS $\mathfrak{S} = (\mathbb{N}^n, \xrightarrow{F}, \leq)$, the procedure $\mathbf{Clover}_{\mathfrak{S}}(s_0)$ terminates.*

6.2 Clover-Flattable Complete WSTS

We now characterize those ∞-effective complete WSTS on which $\mathbf{Clover}_{\mathfrak{S}}$ terminates.

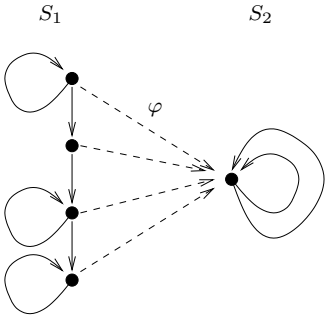

Fig. 4. Flattening

Not all systems of interest are flat. The simplest example of a non-flat system has one state q and two transitions $q \xrightarrow{a} q$ and $q \xrightarrow{b} q$.

For an arbitrary system S, *flattening* [BFLS05] consists in finding a flat system S', equivalent to S with respect to reachability, and in computing on S' instead of S. We adapt the definition in [BFLS05] to functional transition systems, without an explicit finite control graph for now (but see Definition 11).

Definition 7 (Flattening). *A* flattening *of a functional transition system $\mathfrak{S}_2 = (S_2, \xrightarrow{F_2})$ is a pair $(\mathfrak{S}_1, \varphi)$, where:*

1. *$\mathfrak{S}_1 = (S_1, \xrightarrow{F_1})$ is a* flat *functional transition system;*
2. *and $\varphi : \mathfrak{S}_1 \to \mathfrak{S}_2$ is a* morphism *of transition systems. That is, φ is a pair of two maps, both written φ, from S_1 to S_2 and from F_1 to F_2, such that for all $(s, s') \in S_1^2$, for all $f_1 \in F_1$ such that $s \in \operatorname{dom} f_1$ and $s' = f_1(s)$, $\varphi(s) \in \operatorname{dom} \varphi(f_1)$ and $\varphi(s') = \varphi(f_1)(\varphi(s))$ (see Figure 4).*

Let us recall that a pair $(\mathfrak{S}, s_0)$ of a transition system and a state is *$Post^*$-flattable* iff there is a flattening $\mathfrak{S}_1$ of $\mathfrak{S}$ and a state s_1 of $\mathfrak{S}_1$ such that $\varphi(s_1) = s_0$ and $Post^*_{\mathfrak{S}}(s_0) = \varphi(Post^*_{\mathfrak{S}_1}(s_1))$.

Recall that we equate ordered functional transition systems $(S, \xrightarrow{F}, \leq)$ with their underlying function transition system $(S, \xrightarrow{F})$. The notion of flattening then extends to ordered functional transition systems. However, it is then natural to consider *monotonic flattenings*, where in addition $\varphi : S_1 \to S_2$ is monotonic. In the case of complete transition systems, the natural extension requires φ to be continuous:

Definition 8 (Continuous Flattening). *Let $\mathfrak{S}_2 = (S_2, \xrightarrow{F_2}, \leq_2)$ be a complete transition system. A flattening $(\mathfrak{S}_1, \varphi)$ of $\mathfrak{S}_2$ is* continuous *iff:*

1. *$\mathfrak{S}_1 = (S_1, \xrightarrow{F_1}, \leq_1)$ is a* complete *transition system;*
2. *and $\varphi : S_1 \to S_2$ is* continuous*.*

Definition 9 (Clover-Flattable). *Let $\mathfrak{S}$ be a complete transition system, and s_0 be a state. We say that $(\mathfrak{S}, s_0)$ is* clover-flattable *iff there is an continuous flattening $(\mathfrak{S}_1, \varphi)$ of $\mathfrak{S}$, and a state s_1 of $\mathfrak{S}_1$ such that:*

1. *$\varphi(s_1) = s_0$ (φ maps initial states to initial states);*
2. *$cl(Cover_{\mathfrak{S}}(s_0)) = cl(\varphi\langle cl(Cover_{\mathfrak{S}_1}(s_1))\rangle)$ (φ preserves the closures of the covers of the initial states).*

On complete WSTS—our object of study—, the second condition can be simplified to $\downarrow Clover_{\mathfrak{S}}(s_0) = \downarrow\varphi(Clover_{\mathfrak{S}_1}(s_1))$ (using Proposition 1 and the fact that φ, as a continuous map, is monotonic), or equivalently to $Clover_{\mathfrak{S}}(s_0) = \operatorname{Max}\varphi\langle Clover_{\mathfrak{S}_1}(s_1)\rangle$. Recall also that, when $\mathfrak{S}$ is the completion $\widehat{\mathfrak{X}}$ of a WSTS $\mathfrak{X} = (X, \xrightarrow{F}, \leq)$, the clover of $s_0 \in X$ is a finite description of the *cover* of s_0 in $\mathfrak{X}$ (Proposition 2), and this is what φ should preserve, up to taking downward closures.

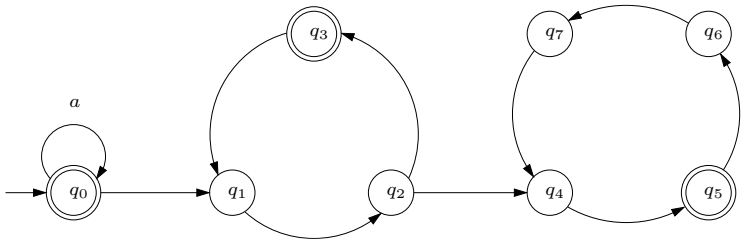

Fig. 5. An rl-automaton

Let us define the synchronized product.

Definition 10 (Synchronized Product). *Let $\mathfrak{S} = (S, \xrightarrow{F}, \leq)$ be a complete functional transition system, and $\mathcal{A} = (F, Q, \delta, q_0)$ be an rl-automaton on the same alphabet F.*

Define the synchronized product *$\mathfrak{S} \times \mathcal{A}$ as the ordered functional transition system $(S \times Q, \xrightarrow{F'}, \leq')$, where F' is the collection of all partial maps $f \bowtie \delta : (s, q) \mapsto (f(s), \delta(q, f))$, for each $f \in F$ such that $\delta(q, f)$ is defined for some $q \in Q$. Let also $(s, q) \leq' (s', q')$ iff $s \leq s'$ and $q = q'$.*

Let π_1 be the morphism of transition systems defined as first projection on states; i.e., $\pi_1(s, q) = s$ for all $(s, q) \in S \times Q$, $\pi_1(f \bowtie \delta) = f$ for all $f \in F$.

Lemma 3 (Synchronized Product). *Let $\mathfrak{S} = (S, \xrightarrow{F}, \leq)$ be a complete functional transition system, and $\mathcal{A} = (F, Q, \delta, q_0)$ be an rl-automaton on the same alphabet F.*

Then $(\mathfrak{S} \times \mathcal{A}, \pi_1)$ is a continuous flattening of $\mathfrak{S}$.

Strong flattenings are special: the decision to take the next action $f \in F$ from state (s, q) is dictated by the current control state q *only*, while ordinary flattenings allow more complex decisions to be made.

We say that a transition system is strongly clover-flattable iff we can require that the flat system $\mathfrak{S}_1$ is a synchronized product, and the continuous morphism of transition systems φ is first projection π_1:

Definition 11 (Strongly Clover-Flattable). *Let $\mathfrak{S} = (S, \xrightarrow{F})$ be a complete functional transition system. We say that $(\mathfrak{S}, s_0)$ is* strongly clover-flattable *iff there is an rl-automaton $\mathcal{A}$, say with initial state q_0, such that $cl(Cover_{\mathfrak{S}}(s_0)) = cl(\pi_1\langle cl(Cover_{\mathfrak{S}\times\mathcal{A}}(s_0, q_0))\rangle)$.*

The following is then obvious.

Lemma 4. *On complete functional transition systems, the implications "strongly clover-flattable" $\Longrightarrow$ "clover-flattable" $\Longrightarrow$ "weakly clover-flattable" hold.*

It is also easy to show that "weakly clover-flattable" also implies "clover-flattable". However, we shall show something more general in Theorem 7 below.

We show in Proposition 8 that $\mathbf{Clover}_{\mathfrak{S}}(s_0)$ can only terminate when $(\mathfrak{S}, s_0)$ is strongly clover-flattable. We shall require the following lemma. For notational simplicity, we equate words $g_1 g_2$ with compositions $g_2 \circ g_1$.

Lemma 5. *Let $\mathfrak{S} = (S, \xrightarrow{F})$ be a complete functional transition system, and $s_0 \in F$. Assume ${g_1}^\infty {g_2}^\infty \dots {g_n}^\infty(s_0)$ is defined, and in some open subset U of S, for some $g_1, g_2, \dots, g_n \in F$. Then there are natural numbers $k_1, k_2, \dots, k_n$ such that $g_1^{k_1} g_2^{k_2} \dots g_n^{k_n}(s_0)$ is defined, and in U.*

Proposition 8. *Let $\mathfrak{S}$ be an ∞-effective complete WSTS. If $\mathbf{Clover}_{\mathfrak{S}}$ terminates on s_0, then $(\mathfrak{S}, s_0)$ is strongly clover-flattable.*

We now loop the loop and show that $\mathbf{Clover}_{\mathfrak{S}}$ terminates on s_0 whenever $(\mathfrak{S}, s_0)$ is weakly clover-flattable (Theorem 7 below). This may seem obvious. In

particular, if $(\mathfrak{S}, s_0)$ is clover-flattable, then accelerate along the loops from $\mathfrak{S}_1$, where $\mathfrak{S}_1$, φ is a continuous flattening of $\mathfrak{S}$. The difficulty is that we *cannot* actually choose to accelerate whenever we want: the $\mathbf{Clover}_{\mathfrak{S}}$ procedure decides by itself when it should accelerate, independently of any flattening whatsoever.

There is an added difficulty, in the sense that one should also check that lub-accelerations, as they are used in $\mathbf{Clover}_{\mathfrak{S}}$, are enough to reach all required least upper bounds. The key point is the following lemma, which asserts the existence of finitely many subsequences $g^{p_j+\ell q_j}(s)$, $\ell \in \mathbb{N}$, whose exponents form infinite arithmetic progressions, and which generate all possible limits of directed families of elements of the form $g^n(s)$, $n \in \mathbb{N}$, except possibly for finitely many isolated points.

This is the point in our study where progress is needed. Indeed, we require S to be wpo to pick k and m in the proof below.

Lemma 6. *Let S be a dcwo, $g : S \to S$ a partial monotonic map, and $s \in S$. Consider the family G of all elements of the form $g^n(s)$, for those $n \in \mathbb{N}$ such that this is defined. Then there are finitely many directed subfamilies G_0, G_1, ..., G_{m-1} of G such that:*

1. *$cl(G) = \bigcup_{j=0}^{m-1} cl(G_j) = {\downarrow}\{\mathrm{lub}(G_0), \mathrm{lub}(G_1), \ldots, \mathrm{lub}(G_{m-1})\}$;*
2. *each G_j is either a one-element set $\{g^{p_j}(s)\}$, where $p_j \in \mathbb{N}$, or is a chain of the form $\{g^{p_j+\ell q_j}(s) \mid \ell \in \mathbb{N}\}$, where $p_j \in \mathbb{N}$, $q_j \in \mathbb{N} \setminus \{0\}$, and $g^{p_j}(s) < g^{p_j+q_j}(s)$;*
3. *for every j, $0 \leq j < m$, $s \not\leq g^{p_j}(s)$.*

Proposition 9. *Let $\mathfrak{S}$ be an ∞-effective complete WSTS. Assume that $(\mathfrak{S}, s_0)$ is weakly clover-flattable. Then $\mathbf{Clover}_{\mathfrak{S}}$ terminates on s_0.*

Putting together Lemma 4, Proposition 8, and Proposition 9, we obtain:

Theorem 7 (Main Theorem). *Let $\mathfrak{S}$ be an ∞-effective complete WSTS. The following statements are equivalent:*

1. *$(\mathfrak{S}, s_0)$ is clover-flattable;*
2. *$(\mathfrak{S}, s_0)$ is weakly clover-flattable;*
3. *$(\mathfrak{S}, s_0)$ is strongly clover-flattable;*
4. *$\mathbf{Clover}_{\mathfrak{S}}(s_0)$ terminates.* □

6.3 Cover-flattability (without the "l" in "Cover")

Turning to non-complete WSTS, we define:

Definition 12 (Monotonic Flattening). *Let $\mathfrak{X}_2 = (X_2, \overset{F_2}{\to}, \leq_2)$ be an ordered functional transition system. A flattening $(\mathfrak{X}_1, \varphi)$ of $\mathfrak{X}_2$ is* monotonic *iff:*

1. *$\mathfrak{X}_1 = (X_1, \overset{F_1}{\to}, \leq_1)$ is an ordered functional transition system;*
2. *and $\varphi : X_1 \to X_2$ is* monotonic.

Definition 13 (Cover-Flattable). *Let $\mathfrak{X}$ be an ordered functional transition system, and x_0 be a state. We say that $(\mathfrak{X}, x_0)$ is* cover-flattable *iff there is a monotonic flattening $(\mathfrak{X}_1, \varphi)$ of $\mathfrak{X}$, and a state x_1 of $\mathfrak{X}_1$ such that:*

1. $\varphi(x_1) = x_0$;
2. $Cover_{\mathfrak{X}}(x_0) = {\downarrow}\varphi\langle Cover_{\mathfrak{X}_1}(x_1)\rangle$.

Theorem 8. *Let $\mathfrak{X} = (X, \xrightarrow{F}, \leq)$ be an ω^2-WSTS that is ∞-effective, in the sense that $\widehat{\mathfrak{X}}$ is ∞-effective, i.e., that $(\mathcal{S}g)^\infty$ is computable for every $g \in F^*$. The following statements are equivalent:*

1. *$(\mathfrak{X}, x_0)$ is cover-flattable;*
2. *$(\widehat{\mathfrak{X}}, \eta_X(x_0))$ is (weakly, strongly) clover-flattable;*
3. ***Clover**$_{\widehat{\mathfrak{X}}}(\eta_X(x_0))$ terminates.*

In this case, **Clover**$_{\widehat{\mathfrak{X}}}(\eta_X(x_0))$ *returns the clover $A = Clover_{\mathfrak{S}}(s_0)$, and this is a finite description of the cover, in the sense that $Cover_{\mathfrak{X}}(x_0) = \eta_X^{-1}({\downarrow}A)$.*

By a slight abuse of language, say that a functional WSTS $\mathfrak{S} = (S, \xrightarrow{F}, \leq)$ is cover-flattable iff $(\mathfrak{S}, s_0)$ is cover-flattable for every initial state $s_0 \in S$.

Corollary 1. *Every Petri net, and every VASS, is cover-flattable.*

Proof. The state space of a Petri net on k places is $\mathbb{N}^k$, that of a VASS [HP79] is $Q \times \mathbb{N}^k$, where Q is a finite set of control states. We deal with the latter, as they are more general. Transitions of the VASS $\mathfrak{X}$ are of the form $f(q, \boldsymbol{x}) = (q', \boldsymbol{x} + \boldsymbol{b} - \boldsymbol{a})$, provided $\boldsymbol{x} \geq \boldsymbol{a}$, and where $\boldsymbol{a}$, $\boldsymbol{b}$ are fixed tuples in $\mathbb{N}^k$. It is easy to see that $\mathcal{S}f$ is defined by: $\mathcal{S}f(q, \boldsymbol{x}) = (q', \boldsymbol{x} + \boldsymbol{b} - \boldsymbol{a})$, provided $\boldsymbol{x} \geq \boldsymbol{a}$, this time for all $\boldsymbol{x} \in \mathbb{N}_\omega^k$. So the completion $\widehat{\mathfrak{S}}$ of the VASS is ∞-effective. On these, the Karp-Miller algorithm terminates [KM69], hence also the generalized Karp-Miller algorithm of Section 5.1. By Proposition 6, **Clover**$_{\widehat{\mathfrak{S}}}$ terminates on any input $s_0 \in Q \times \mathbb{N}_\omega^k$. So $\mathfrak{X}$ is cover-flattable, by Theorem 8. □

Corollary 2. *There are reset Petri nets, and functional-lossy channel systems that are not cover-flattable.*

7 Well Structured Presburger Counter Systems

We now demonstrate how the fairly large class of counter systems fits with our theory. We show that counter systems composed of affine monotonic functions with upward-closed definition domains are complete (strongly monotonic) WSTS. This result is obtained by showing that every monotonic affine function f is continuous and its lub-acceleration f^∞ is computable [CFS11]. Moreover, we prove that it is possible to decide whether a general counter system (given by a finite set of Presburger relations) is a monotonic affine counter system, but that one cannot decide whether it is a WSTS.

Definition 14. *A* Presburger counter system *(with n counters), $\mathcal{C}$ is a tuple $\mathcal{C} = (Q, R, \to)$ where Q is a finite set of control states, $R = \{r_1, r_2, ...r_k\}$ is a finite set of Presburger relations $r_i \subseteq \mathbb{N}^n \times \mathbb{N}^n$ and $\to \subseteq Q \times R \times Q$.*

We will consider a special case of Presburger relations, those which allow us to encode the graph of affine functions. A (partial) function $f : \mathbb{N}^n \longrightarrow \mathbb{N}^n$ is *non-negative affine*, for short *affine* if there exist a matrix $A \in \mathbb{N}^{n \times n}$ with *non-negative coefficients* and a vector $b \in \mathbb{Z}^n$ such that for all $\boldsymbol{x} \in \operatorname{dom} f, f(\boldsymbol{x}) = A\boldsymbol{x} + \boldsymbol{b}$. When necessary, we will extend affine maps $f : \mathbb{N}^n \longrightarrow \mathbb{N}^n$ by continuity to $f : \mathbb{N}^n_\omega \longrightarrow \mathbb{N}^n_\omega$, by $f(\operatorname{lub}_{i \in \mathbb{N}}(\boldsymbol{x}_i)) = \operatorname{lub}_{i \in \mathbb{N}}(f(\boldsymbol{x}_i))$ for every countable chain $(\boldsymbol{x}_i)_{i \in \mathbb{N}}$ in $\mathbb{N}^n$. That is, we just write f instead of $\mathcal{S}f$.

Definition 15. *An* Affine Counter System *(with n counters), a.k.a. an* ACS *$\mathcal{C} = (Q, R, \to)$ is a Presburger counter system where all relations r_i are (partial) affine functions.*

The domain of maps f in an affine counter system ACS are Presburger-definable. A reset/transfer Petri net is an ACS where every line or column of every matrix contains at most one non-zero coefficient equal to 1, and, all domains are upward-closed sets. A *Petri net* is an ACS where all affine maps are translations with upward-closed domains.

Theorem 9. *One can decide whether an effective Presburger counter system is an ACS.*

Proof. The formula expressing that a relation is a function is a Presburger formula, hence one can decide whether R is the graph of a function. One can also decide whether the graph G_f of a function f is monotonic because monotonicity of a Presburger-definable function can be expressed as a Presburger formula. Finally, one can also decide whether a Presburger formula represents an affine function $f(\boldsymbol{x}) = A\boldsymbol{x} + \boldsymbol{b}$ with $A \in \mathbb{N}^{n \times n}$ and $\boldsymbol{b} \in \mathbb{Z}^n$, using results by Demri *et al.* [DFGvD06]. □

For counter systems (which include Minsky machines), monotonicity is undecidable. Clearly, a counter system $\mathfrak{S}$ is well-structured iff $\mathfrak{S}$ is monotonic: so there is no algorithm to decide whether a Presburger counter system is a WSTS. However, an ACS is strongly monotonic iff each map f is partial monotonic; this is equivalent to requiring that $\operatorname{dom} f$ is upward-closed, since all matrices A have non-negative coefficients. This is easily cast as Presburger formula, and therefore decidable.

Proposition 10. *There is an algorithm to decide whether an ACS is a strongly monotonic WSTS.*

Proof. The strong monotony of an ACS $\mathcal{C}$ means that every function of $\mathcal{C}$ is monotonic and this can be expressed by a Presburger formula saying that all the (Presburger-definable) definition domains are upward-closed (the matrices are known to be positive). □

We have recalled that the transitions function of Petri nets ($f(x) = x + \boldsymbol{b}$, $\boldsymbol{b} \in \mathbb{Z}^n$ and $\mathrm{dom}(f)$ upward-closed) can be lub-accelerated effectively. This result was generalized to broadcast protocols (equivalent to transfer Petri nets) by Emerson and Namjoshi [EN98] and to another class of monotonic affine functions $f(\boldsymbol{x}) = A\boldsymbol{x}+\boldsymbol{b}$ such that $A \in \mathbb{N}^{n\times n}$, $b \in \mathbb{N}^n$ (note that b is not in $\mathbb{Z}^n$) and $\mathrm{dom}(f)$ is upward closed [FMP04].

[CFS11] recently extended this result to all monotonic affine functions: for every $f(\boldsymbol{x}) = A\boldsymbol{x} + \boldsymbol{b}$ with $A \in \mathbb{N}^{n\times n}$, $\boldsymbol{b} \in \mathbb{Z}^n$ and $\mathrm{dom}(f)$ upward-closed, the function f^∞ is recursive.

We deduce the following strong relationship between well-structured ACS and complete well-structured ACS.

Theorem 10. *The completion of an ACS S is an ∞-effective complete WSTS iff S is a strongly monotonic WSTS.*

Proof. Strong monotonicity reduces to partial monotonicity of each map f, as discussed above. Well-structured ACS are clearly effective, since $Post(\boldsymbol{s}) = \{\boldsymbol{t} \mid \exists f \in F \cdot f(\boldsymbol{t}) = \boldsymbol{s}\}$ is Presburger-definable. Note also that monotonic affine function are continuous, and $\mathbb{N}^n_\omega$ is a continuous dcwo. Finally, for every Presburger monotonic affine function f, the function f^∞ is recursive, so the considered ACS is ∞-effective. □

Corollary 3. *One can decide whether the completion of an ACS is an ∞-effective complete WSTS.*

So the completions of reset/transfer Petri nets [DFS98], broadcast protocols [EFM99], self-modifying Petri nets [Val78] and affine well-structured nets [FMP04] are ∞-effective complete WSTS.

8 Conclusion and Perspectives

We have provided a framework of *complete WSTS*, and of *completions* of WSTS, on which forward reachability analyses can be conducted, using natural finite representations for downward-closed sets. The central element of this theory is the *clover*, i.e., the set of maximal elements of the closure of the cover. We have shown that, for complete WSTS, the clover is finite and describes the closure of the cover exactly. When the original WSTS is not complete,

We have also defined a simple procedure, $\mathbf{Clover}_{\mathfrak{S}}$ for computing the clover for ∞-effective complete WSTS $\mathfrak{S}$. This captures the essence of generalized forms of the Karp-Miller procedure, while terminating in more cases. We have shown that that $\mathbf{Clover}_{\mathfrak{S}}$ terminates iff the WSTS is *clover-flattable*, i.e., that it is some form of projection of a flat system, with the same clover. We have also shown that several variants of the notion of clover-flattability were in fact equivalent. We believe that this characterization is an important, and non-trivial result.

In the future, we shall explore efficient strategies for choosing sequences $g \in F^*$ to lub-accelerate in the $\mathbf{Clover}_{\mathfrak{S}}$ procedure. We will also analyze whether

Clover$_{\mathfrak{S}}$ terminates in models such as BVASS [VG05], reconfigurable nets, timed Petri nets [ADMN04], post-self-modifying Petri nets [Val78] and strongly monotonic affine well-structured nets [FMP04]), i.e., whether they are cover-flattable.

One potential use of the clover is in deciding coverability. But the **Clover**$_{\mathfrak{S}}$ procedure may fail to terminate. This is in contrast to the Expand, Enlarge and Check forward algorithm of [GRvB07], which always terminates, hence decides coverability. One may want to combine the best of both worlds, and the lub-accelerations of **Clover**$_{\mathfrak{S}}$ can profitably be used to improve the efficiency of the Expand, Enlarge and Check algorithm. This remains to be explored.

Finally, recall that computing the finite clover is a first step [EN98] in the direction of solving liveness properties (and not only safety properties which reduce to coverability). We plan to clarify the construction of a *cloverability graph* which would be the basis for liveness model checking.

Acknowledgement

We wish to acknowledge Rémi Bonnet and Sylvain Schmitz for fruitful discussions.

References

[AJ94] Parosh Aziz Abdulla and Bengt Jonsson. Undecidable Verification Problems for Programs with Unreliable Channels. In ICALP'94, pp. 346-327.

[ABJ98] Parosh Abdulla, Ahmed Bouajjani, and Bengt Jonsson. On-the-fly analysis of systems with unbounded, lossy Fifo channels. In *Proc. 10th Intl. Conf. Computer Aided Verification (CAV'98)*, pages 305–318, Vancouver, Canada, June 1998. Springer Verlag LNCS 1427.

[ADB07] Parosh Aziz Abdulla, Giorgio Delzanno and Laurent Van Begin. Comparing the Expressive Power of Well-Structured Transition Systems. In CSL, pp. 99-114, 2007.

[AN00] Parosh Abdulla and Aletta Nylén. Better is Better than Well: On Efficient Verification of Infinite-State Systems. 14th LICS, pp. 132-140, 2000.

[ACABJ04] Parosh Aziz Abdulla, Aurore Collomb-Annichini, Ahmed Bouajjani, and Bengt Jonsson. Using forward reachability analysis for verification of lossy channel systems. *Formal Methods in System Design*, 25(1):39–65, 2004.

[AČJT00] Parosh Aziz Abdulla, Karlis Čerāns, Bengt Jonsson, and Yih-Kuen Tsay. Algorithmic analysis of programs with well quasi-ordered domains. *Information and Computation*, 160(1–2):109–127, 2000.

[ADMN04] Parosh Aziz Abdulla, Johann Deneux, Pritha Mahata, and Aletta Nylén. Forward reachability analysis of timed Petri nets. In *FORMATS/FTRTFT*, pages 343–362. Springer Verlag LNCS 3253, 2004.

[AJ94] Samson Abramsky and Achim Jung. Domain theory. In S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 3, pages 1–168. Oxford University Press, 1994.

[BF12] Rémi Bonnet and Alain Finkel. Forward Analysis for WSTS : Beyond Regular Accelerations. Submitted, february 2012.

[BFLS05] Sébastien Bardin, Alain Finkel, Jérôme Leroux, and Philippe Schnoebelen. Flat acceleration in symbolic model checking. In *Proc. 3rd Intl. Symp. Automated Technology for Verification and Analysis (ATVA'05)*, pages 474–488. Springer Verlag LNCS 3707, 2005.

[BFHRV11] Rémi Bonnet, Alain Finkel, Serge Haddad and Fernando Rosa-Velardo. Ordinal Theory for Expressiveness of Well Structured Transition Systems. In Proceedings of the 14th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS'11), Saarbrücken, Germany, March-April 2011, LNCS 6604, pages 153-167. Springer.

[BG11] Laura Bozzelli and Pierre Ganty. Complexity analysis of the backward coverability algorithm for vass. In *Proceedings of the 5th international conference on Reachability problems*, RP'11, pages 96–109, Berlin, Heidelberg, 2011. Springer-Verlag.

[CFP96] Gérard Cécé, Alain Finkel, and S. Purushothaman Iyer. Unreliable channels are easier to verify than perfect channels. *Information and Computation*, 124(1):20–31, January 1996.

[CFS11] Pierre Chambart, Alain Finkel, and Sylvain Schmitz. Forward analysis and model checking for trace bounded WSTS. In Lars M. Kristensen and Laure Petrucci, editors, *Proceedings of the 32nd International Conference on Applications and Theory of Petri Nets (ICATPN'11)*, volume 6709 of *Lecture Notes in Computer Science*, Newcastle upon Tyne, UK, June 2011. Springer.

[CFS11] Pierre Chambart , Alain Finkel and Sylvain Schmitz. Forward Analysis and Model Checking for Trace Bounded WSTS. In Petri Nets, pp. 49-68, 2011.

[DFGvD06] Stéphane Demri, Alain Finkel, Valentin Goranko, and Govert van Drimmelen. Towards a model-checker for counter systems. In *4th ATVA*, pages 493–507. Springer Verlag LNCS 4218, 2006.

[DFS98] Catherine Dufourd, Alain Finkel, and Philippe Schnoebelen. Reset nets between decidability and undecidability. In *Proc. 25th Intl. Coll. Automata, Languages and Programming (ICALP'98)*, pages 103–115. Springer Verlag LNCS 1443, 1998.

[EFM99] Javier Esparza, Alain Finkel, and Richard Mayr. On the verification of broadcast protocols. In *14th LICS*, pages 352–359, 1999.

[EN98] E. Allen Emerson and Kedar S. Namjoshi. On model-checking for nondeterministic infinite-state systems. In *13th LICS*, pages 70–80, 1998.

[FFSS11] Diego Figueira and Santiago Figueira and Sylvain Schmitz and Philippe Schnoebelen. Ackermannian and Primitive-Recursive Bounds with Dickson's Lemma. In LICS, pp. 269-278, 2011.

[Fin87] Alain Finkel. A generalization of the procedure of Karp and Miller to well structured transition systems. In *Proc. 13th Intl. Coll. Automata, Languages and Programming (ICALP'87)*, pages 499–508. Springer Verlag LNCS 267, 1987.

[Fin90] Alain Finkel. Reduction and covering of infinite reachability trees. *Information and Computation*, 89(2):144–179, 1990.

[Fin93] Alain Finkel. The minimal coverability graph for Petri nets. In *12th Intl. Conf. Advances in Petri Nets*, pages 210–243. Springer Verlag LNCS 674, 1993.

[FMP04] Alain Finkel, Pierre McKenzie, and Claudine Picaronny. A well-structured framework for analysing Petri net extensions. *Information and Computation*, 195(1-2):1–29, 2004.

[FS01] Alain Finkel and Philippe Schnoebelen. Well-structured transition systems everywhere! *Theoretical Computer Science*, 256(1–2):63–92, 2001.

[FG09a] Alain Finkel and Jean Goubault-Larrecq. Forward analysis for WSTS, part I: Completions. In Susanne Albers and Jean-Yves Marion, editors, *Proceedings of the 26th Annual Symposium on Theoretical Aspects of Computer Science (STACS'09)*, volume 3 of *Leibniz International Proceedings in Informatics*, pages 433–444, Freiburg, Germany, February 2009. Leibniz-Zentrum für Informatik.

[FG09b] Alain Finkel and Jean Goubault-Larrecq. Forward analysis for WSTS, Part II: Complete WSTS. In S. Albers, A. Marchetti-Spaccamela, Y. Matias, S. E. Nikoletseas, and W. Thomas, editors, *36th Int. Colloquium on Automata, Languages and Programming, ICALP'09*, volume 5556 of *Lect. Notes Comp. Sci.*, pages 188–199. Springer, 2009.

[FG12a] Alain Finkel and Jean Goubault-Larrecq. Forward analysis for WSTS, part I: Completions. In preparation, 2012. Journal version of [FG09a].

[FG12b] Alain Finkel and Jean Goubault-Larrecq. Forward analysis for WSTS, Part II: Complete WSTS. In preparation, 2012. Journal version of [FG09b].

[GRVB07] Gilles Geeraerts, Jean-François Raskin and Laurent Van Begin. Well-structured languages. In Acta Inf., volume 44, number 3-4, pp. 249-288, 2007.

[GHK+03] Gerhard Gierz, Karl Heinrich Hofmann, Klaus Keimel, Jimmie D. Lawson, Michael Mislove, and Dana S. Scott. Continuous lattices and domains. In *Encyclopedia of Mathematics and its Applications*, volume 93. Cambridge University Press, 2003.

[GRvB06a] Pierre Ganty, Jean-François Raskin, and Laurent van Begin. A complete abstract interpretation framework for coverability properties of WSTS. In E. Allen Emerson and Kedar S. Namjoshi, editors, *Proc. 7th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI'06)*, pages 49–64. Springer Verlag LNCS 3855, 2006.

[GRvB06b] Gilles Geeraerts, Jean-François Raskin, and Laurent van Begin. Expand, enlarge and check: New algorithms for the coverability problem of WSTS. *J. Comp. and System Sciences*, 72(1):180–203, 2006.

[GRvB07] Gilles Geeraerts, Jean-François Raskin, and Laurent van Begin. On the efficient computation of the minimal coverability set for Petri nets. In *Proc. 5th Intl. Symp. Automated Technology for Verification and Analysis (ATVA'05)*, pages 98–113. Springer LNCS 4762, 2007.

[GS64] Seymour Ginsburg and Edwin H. Spanier. Bounded Algol-like languages. *Trans. American Mathematical Society*, 113(2):333–368, 1964.

[HP79] John Hopcroft and Jean-Jacques Pansiot. On the reachability problem for 5-dimensional vector addition systems. *Theoretical Computer Science*, 8:135–159, 1979.

[Jan99] Petr Jančar. A note on well quasi-orderings for powersets. *Information Processing Letters*, 72(5–6):155–160, 1999.

[KM69] Richard M. Karp and Raymond E. Miller. Parallel program schemata. *J. Comp. and System Sciences*, 3(2):147–195, 1969.

[KS96] Olga Kouchnarenko and Ph. Schnoebelen. A model for recursive-parallel programs. In Electr. Notes Theor. Comput. Sci., volume 5, 30 pages, 1996.

[LNORW07] Ranko Lazic, Thomas Christopher Newcomb, Joël Ouaknine, A. W. Roscoe and James Worrell. Nets with Tokens Which Carry Data. In ICATPN, pp. 301-320, 2007.

[Mar94] Alberto Marcone. Foundations of BQO theory. *Trans. Amer. Math. Soc.*, 345(2):641–660, 1994.

[May03a] Richard Mayr. Undecidable problems in unreliable computations. *Theor. Comput. Sci.*, 297(1-3):337–354, 2003.

[May03b] Richard Mayr. Undecidable problems in unreliable computations. *Theoretical Computer Science*, 297(1-3):337–354, 2003.

[MM81] Ernst W. Mayr and Albert R. Meyer. The complexity of the finite containment problem for petri nets. *J. ACM*, 28(3):561–576, 1981.

[Rac78] Charles Rackoff. The covering and boundedness problems for vector addition systems. *Theor. Comput. Sci.*, 6:223–231, 1978.

[RMF11] Fernando Rosa-Velardo and María Martos-Salgado and David de Frutos-Escrig. Accelerations for the Coverability Set of Petri Nets with Names. In Fundam. Inform., volume 113, number 3-4, pp. 313-341, 2011.

[RS04] Neil Robertson and P.D. Seymour. Graph minors. XX. Wagner's conjecture. *Journal of Combinatorial Theory, Series B*, 92(2):325–357, 2004.

[Sch01] Philippe Schnoebelen. Bisimulation and other undecidable equivalences for lossy channel systems. In Naoki Kobayashi and Benjamin C. Pierce, editors, *Proceedings of the 4th International Workshop on Theoretical Aspects of Computer Software (TACS'01)*, volume 2215 of *Lecture Notes in Computer Science*, pages 385–399, Sendai, Japan, October 2001. Springer.

[SS11] Sylvain Schmitz and Ph. Schnoebelen. Multiply-Recursive Upper Bounds with Higman's Lemma. In ICALP (2), pp. 441-452, 2011.

[Val78] Rüdiger Valk. Self-modidying nets, a natural extension of Petri nets. In *Proceedings of the 5th International Colloquium on Automata, Languages and Programming (ICALP'78)*, pages 464–476. Springer Verlag LNCS 62, 1978.

[VF07] Fernando Rosa-Velardo and David de Frutos-Escrig. Name Creation vs. Replication in Petri Net Systems. In ICATPN, pp. 402-422, 2007.

[VG05] Kumar N. Verma and Jean Goubault-Larrecq. Karp-Miller trees for a branching extension of VASS. *Discrete Mathematics & Theoretical Computer Science*, 7(1):217–230, November 2005.

[WZH10] Thomas Wies, Damien Zufferey, and Thomas A. Henzinger. Forward analysis of depth-bounded processes. In C.-H. Luke Ong, editor, *FOSSACS*, volume 6014 of *Lecture Notes in Computer Science*, pages 94–108. Springer, 2010.

[ZWH12] Damien Zufferey, Thomas Wies, and Thomas A. Henzinger. Ideal abstractions for well-structured transition systems. In *VMCAI*, pages 445–460, 2012.